```
call void @llvm.dbg.value(metadata ptr %a, metadata !21, metadata
                              ... !DIExpression()), !dbg !27
                              call void @llvm.dbg.value(metadata ptr %b, metadata !22, metadata
                              ... !DIExpression()), !dbg !27
                              call void @llvm.dbg.value(metadata ptr %c, metadata !23, metadata
                              ...!DIExpression()),!dbg!27
                              call void @llvm.dbg.value(metadata i32 %n, metadata !24, metadata
                              ... !DIExpression()), !dbg !27
                              call void @llvm.dbg.value(metadata i32 0, metadata !25, metadata
                              ... !DIExpression()), !dbg !28
                               %cmp11 = icmp sgt i32 %n, 0, !dbg !29
                               br i1 %cmp11, label %for.body.preheader, label %for.cond.cleanup, !dbg !31
                     for.body.preheader:
                      %wide.trip.count = zext i32 %n to i64, !dbg !29
                      br label %for.body.init.1
    for.bodv.init.1:
    call void @llvm.dbg.value(metadata i64 0, metadata !25, metadata
    ...!DIExpression()), !dbg!28
    %rem15.init.1 = and i64 0, 1, !dbg !32
    %cmp1.not.init.1 = icmp eq i64 %rem15.init.1, 0, !dbg !32
    %indvars.iv.next.init.1 = add nuw nsw i64 0, 1, !dbg !35
    %rem15.headerCopy.1.init.1 = and i64 %indvars.iv.next.init.1, 1, !dbg !32
    %cmp1.not.headerCopy.1.init.1 = icmp eq i64 %rem15.headerCopy.1.init.1, 0,
    ... !dbg !32
    %indvars.iv.next.latchCopy.1.init.1 = add nuw nsw i64
    ... %indvars.iv.next.init.1, 1, !dbg !35
    %rem15.headerCopy.1.2.init.1 = and i64 %indvars.iv.next.latchCopy.1.init.1,
    ... 1, !dbg !32
    %cmp1.not.headerCopy.1.2.init.1 = icmp eq i64 %rem15.headerCopy.1.2.init.1,
    ... 0, !dbg !32
    %indvars.iv.next.latchCopy.1.2.init.1 = add nuw nsw i64
    ... %indvars.iv.next.latchCopy.1.init.1, 1, !dbg !35
    %rem15.headerCopy.1.2.3.init.1 = and i64
    ... %indvars.iv.next.latchCopy.1.2.init.1, 1, !dbg !32
    %cmp1.not.headerCopy.1.2.3.init.1 = icmp eq i64 ... %rem15.headerCopy.1.2.3.init.1, 0, !dbg !32
    %indvars.iv.next.latchCopy.1.2.3.init.1 = add nuw nsw i64
    ... %indvars.iv.next.latchCopy.1.2.init.1, 1, !dbg !35
    br label %for.body.init.2
   for.body.init.2:
   call void @llvm.dbg.value(metadata i64 0, metadata !25, metadata
   ...!DIExpression()), !dbg!28
   %rem15.init.2 = and i64 %indvars.iv.next.latchCopy.1.2.3.init.1, 1, !dbg !32 %cmp1.not.init.2 = icmp eq i64 %rem15.init.2, 0, !dbg !32
   %indvars.iv.next.init.2 = add nuw nsw i64
    . %indvars.iv.next.latchCopy.1.2.3.init.1, 1, !dbg !35
   %rem15.headerCopy.1.init.2 = and i64 %indvars.iv.next.init.2, 1, !dbg !32
   %cmp1.not.headerCopy.1.init.2 = icmp eq i64 %rem15.headerCopy.1.init.2, 0,
   ...!dbg!32
   %indvars.iv.next.latchCopy.1.init.2 = add nuw nsw i64
   ... %indvars.iv.next.init.2, 1, !dbg !35
   %rem15.headerCopy.1.2.init.2 = and i64 %indvars.iv.next.latchCopy.1.init.2,
   ... 1, !dbg !32
   %cmp1.not.headerCopy.1.2.init.2 = icmp eq i64 %rem15.headerCopy.1.2.init.2,
   .. 0, !dbg !32
   %indvars.iv.next.latchCopy.1.2.init.2 = add nuw nsw i64
   ... %indvars.iv.next.latchCopy.1.init.2, 1, !dbg !35
   %rem15.headerCopy.1.2.3.init.2 = and i64
   ... %indvars.iv.next.latchCopy.1.2.init.2, 1, !dbg !32
   %cmp1.not.headerCopy.1.2.3.init.2 = icmp eq i64
... %rem15.headerCopy.1.2.3.init.2, 0, !dbg !32
   %indvars.iv.next.latchCopy.1.2.3.init.2 = add nuw nsw i64
   ... %indvars.iv.next.latchCopy.1.2.init.2, 1, !dbg !35
   %0 = insertelement <vscale x 4 x i1> undef, i1 %cmp1.not.init.1, i64 0
   %1 = insertelement <vscale x 4 x i1> %0, i1 %cmp1.not.headerCopy.1.init.1,
   ... i64 1
   %2 = insertelement <vscale x 4 x i1> %1, i1 %cmp1.not.headerCopy.1.2.init.1,
   ... i64 2
   %3 = insertelement < vscale x 4 x i1 > %2, i1
   ... %cmp1.not.headerCopy.1.2.3.init.1, i64 3
   %4 = insertelement <vscale x 4 x i1> undef, i1 %cmp1.not.init.2, i64 0
   %5 = insertelement <vscale x 4 x i1> %4, i1 %cmp1.not.headerCopy.1.init.2,
   %6 = insertelement <vscale x 4 x i1> %5, i1 %cmp1.not.headerCopy.1.2.init.2,
   ... i64 2
   \%7 = insertelement <vscale x 4 x i1> \%6, i1
   ... %cmp1.not.headerCopy.1.2.3.init.2, i64 3
   \%8 = \text{call} < \text{vscale x 4 x i32} > \text{@llvm.aarch64.sve.index.nxv4i32(i32 0, i32 1)}
   \%9 = \text{call} < \text{vscale x 4 x i32} > \text{@llvm.aarch64.sve.index.nxv4i32(i32 4, i32 1)}
   br label %for.body
for.body:
%10 = phi i64 [ %indvars.iv.next.latchCopy.1.2.3, %new.latch ], [ 0,
... %for.body.init.2 ]
%11 = phi <vscale x 4 x i32> [ %70, %new.latch ], [ %8, %for.body.init.2 ]
%12 = phi <vscale x 4 x i1> [ %71, %new.latch ], [ %3, %for.body.init.2 ]
%13 = phi <vscale x 4 x i32> [ %72, %new.latch ], [ %9, %for.body.init.2 ]
%14 = phi <vscale x 4 x i1> [ %73, %new.latch ], [ %7, %for.body.init.2 ]
call void @llvm.dbg.value(metadata i64 0, metadata !25, metadata
... !DIExpression()), !dbg !28
%rem15 = and i64 %10, 1, !dbg !32
%cmp1.not = icmp eq i64 %rem15, 0, !dbg !32
%indvars.iv.next = add nuw nsw i64 %10, 1, !dbg !35
%rem15.headerCopy.1 = and i64 %indvars.iv.next, 1, !dbg !32
%cmp1.not.headerCopy.1 = icmp eq i64 %rem15.headerCopy.1, 0, !dbg !32
%indvars.iv.next.latchCopy.1 = add nuw nsw i64 %indvars.iv.next, 1, !dbg !35
%rem15.headerCopy.1.2 = and i64 %indvars.iv.next.latchCopy.1, 1, !dbg !32
%cmp1.not.headerCopy.1.2 = icmp eq i64 %rem15.headerCopy.1.2, 0, !dbg !32 %indvars.iv.next.latchCopy.1.2 = add nuw nsw i64
... %indvars.iv.next.latchCopy.1, 1, !dbg !35
%rem15.headerCopy.1.2.3 = and i64 %indvars.iv.next.latchCopy.1.2, 1, !dbg !32
%cmp1.not.headerCopy.1.2.3 = icmp eq i64 %rem15.headerCopy.1.2.3, 0, !dbg !32
%indvars.iv.next.latchCopy.1.2.3 = add nuw nsw i64
  %indvars.iv.next.latchCopy.1.2, 1, !dbg !35
%15 = sub i64 %wide.trip.count, 3, !dbg !29
%exitcond.not.latchCopy.1.2.3 = icmp eq i64 ... %indvars.iv.next.latchCopy.1.2.3, %15, !dbg !29
br i1 %exitcond.not.latchCopy.1.2.3, label %for.cond.cleanup.loopexit, label
... %permute.decision, !dbg !31, !llvm.loop !37
                                                                                                  permute.decision:
                                                                                                  ^{1}\%16 = \text{call} < \text{vscale x 4 x i1} > \text{@llvm.aarch64.sve.ptrue.nxv4i1(i32 4)}
                                                                                                  %17 = call i64 @llvm.aarch64.sve.cntp.nxv4i1(<vscale x 4 x i1> %16, <vscale
                                                                                                  ... x 4 x i1 > \%12
                                                                                                  %18 = call i64 @llvm.aarch64.sve.cntp.nxv4i1(<vscale x 4 x i1> %16, <vscale
 for.cond.cleanup.loopexit:
                                                                                                   .. \times 4 \times i1 > \%14
 br label %for.cond.cleanup, !dbg !36
                                                                                                  %19 = add i64 %17, %18
                                                                                                  %20 = icmp uge i64 %19, 4
                                                                                                  br i1 %20, label %lane.gather, label %linearized
                                                                                                                                                                F
                                                                            lane.gather:
                                                                            %21 = call <vscale x 4 x i1> @llvm.aarch64.sve.ptrue.nxv4i1(i32 4)
                                                                            %22 = call <vscale x 4 x i32> @llvm.aarch64.sve.compact.nxv4i32(<vscale x 4
                                                                            .. x i1> %12, <vscale x 4 x i32> %11)
                                                                            %23 = call <vscale x 4 x i32> @llvm.aarch64.sve.compact.nxv4i32(<vscale x 4
                                                                            ... x i1> %14, <vscale x 4 x i32> %13)
                                                                            %24 = xor <vscale x 4 x i1> %12, shufflevector (<vscale x 4 x i1>
                                                                            ... insertelement (<vscale x 4 x i1> poison, i1 true, i32 0), <vscale x 4 x i1>
                                                                            ... poison, <vscale x 4 x i32> zeroinitializer)
                                                                            \%25 = xor < vscale x 4 x i1 > \%14, shufflevector (< vscale x 4 x i1 >
                                                                            ... insertelement (<vscale x 4 x i1 > poison, i1 true, i32 0), <vscale x 4 x i1 >
                                                                            ... poison, <vscale x 4 x i32> zeroinitializer)
                                                                            %26 = call <vscale x 4 x i32> @llvm.aarch64.sve.compact.nxv4i32(<vscale x 4
                                                                            ... x i1> %24, <vscale x 4 x i32> %11)
                                                                            %27 = call <vscale x 4 x i32> @llvm.aarch64.sve.compact.nxv4i32(<vscale x 4
                                                                            ... x i1> %25, <vscale x 4 x i32> %13)
                                                                            %28 = call i64 @llvm.aarch64.sve.cntp.nxv4i1(<vscale x 4 x i1> %21, <vscale
                                                                            .. \times 4 \times i1 > \%12
                                                                            %29 = call <vscale x 4 x i1> @llvm.aarch64.sve.whilelt.nxv4i1.i64(i64 0, i64
                                                                            .. %28)
                                                                            %30 = call <vscale x 4 x i32> @llvm.aarch64.sve.splice.nxv4i32(<vscale x 4 x
                                                                            ... i1> %29, <vscale x 4 x i32> %22, <vscale x 4 x i32> %23)
                                                                            %31 = call i64 @llvm.aarch64.sve.cntp.nxv4i1(<vscale x 4 x i1> %21, <vscale
                                                                            .. \times 4 \times i1 > \%14)
                                                                            %32 = call <vscale x 4 x i1> @llvm.aarch64.sve.whilelt.nxv4i1.i64(i64 0, i64
                                                   for.cond.cleanup:
                                                                            ... %31)
                                                   ret void, !dbg !36
                                                                            %33 = call <vscale x 4 x i32> @llvm.aarch64.sve.splice.nxv4i32(<vscale x 4 x
                                                                            ... i1> %32, <vscale x 4 x i32> %23, <vscale x 4 x i32> %27)
                                                                            %34 = call i64 @llvm.aarch64.sve.cntp.nxv4i1(<vscale x 4 x i1> %21, <vscale
                                                                            .. \times 4 \times i1 > \%24
                                                                            %35 = call <vscale x 4 x i1> @llvm.aarch64.sve.whilelt.nxv4i1.i64(i64 0, i64
                                                                            .. %34)
                                                                            %36 = call <vscale x 4 x i32> @llvm.aarch64.sve.sel.nxv4i32(<vscale x 4 x
                                                                            ... i1> %35, <vscale x 4 x i32> %26, <vscale x 4 x i32> %33)
                                                                            %37 = xor < vscale x 4 x i1 > %35, shufflevector (< vscale x 4 x i1 > 
                                                                            ... insertelement (<vscale x 4 x i1> poison, i1 true, i32 0), <vscale x 4 x i1>
                                                                              poison, <vscale x 4 x i32> zeroinitializer)
                                                                            %38 = call i64 @llvm.aarch64.sve.cntp.nxv4i1(<vscale x 4 x i1> %21, <vscale
                                                                            .. \times 4 \times i1 > \%35)
                                                                            %39 = sub i64 %31, %34
                                                                            %40 = add i64 %39, %38
                                                                            %41 = call <vscale x 4 x i1> @llvm.aarch64.sve.whilelt.nxv4i1.i64(i64 0, i64
                                                                            %42 = xor < vscale x 4 x i1 > %41, shufflevector (< vscale x 4 x i1 > %41)
                                                                            ... insertelement (<vscale x 4 x i1> poison, i1 true, i32 0), <vscale x 4 x i1>
                                                                            ... poison, <vscale x 4 x i32> zeroinitializer)
                                                                            %43 = \text{and} < \text{vscale x 4 x i1} > %37, %41
                                                                            %44 = \text{and} < \text{vscale x } 4 \text{ x i1} > \%37, \%41
                                                                            %45 = \text{and} < \text{vscale x } 4 \text{ x } i1 > \%44, \%42
                                                                            %46 = \text{or} < \text{vscale x 4 x i1} > %43, %45
                                                                            br label %if.then
                                                   if.then:
                                                   %arrayidx = getelementptr inbounds i32, ptr %a, i64 %10, !dbg !41
                                                   %arrayidx3 = getelementptr inbounds i32, ptr %b, i64 %10, !dbg !47
                                                   %arrayidx5 = getelementptr inbounds i32, ptr %c, i64 %10, !dbg !49
                                                   %59 = call <vscale x 4 x i1> @llvm.aarch64.sve.ptrue.nxv4i1(i32 4), !dbg !51
                                                    \%60 = \text{call} < \text{vscale x 4 x i32} >
                                                   ... @llvm.aarch64.sve.ld1.gather.sxtw.index.nxv4i32(<vscale x 4 x i1> %59, ptr
                                                                                                                                             linearized:
                                                   ... %arrayidx, <vscale x 4 x i32> %30), !dbg !51
                                                                                                                                             %47 = getelementptr inbounds i32, ptr %a, i64 %10, !dbg !41
                                                                                                                                             %48 = load i32, ptr %47, align 4, !dbg !41, !tbaa !43
                                                   \%61 = \text{call} < \text{vscale x 4 x i32} >
                                                                                                                                             %49 = getelementptr inbounds i32, ptr %b, i64 %10, !dbg !47
                                                   ... @llvm.aarch64.sve.ld1.gather.sxtw.index.nxv4i32(<vscale x 4 x i1> %59, ptr
                                                   ... %arrayidx3, <vscale x 4 x i32> %30), !dbg !51
                                                                                                                                             %50 = load i32, ptr %49, align 4, !dbg !47, !tbaa !43
                                                                                                                                             %51 = mul nsw i32 %50, %48, !dbg !48
                                                   %62 = call <vscale x 4 x i32> @llvm.aarch64.sve.mul.nxv4i32(<vscale x 4 x
                                                   ... i1> %59, <vscale x 4 x i32> %61, <vscale x 4 x i32> %60), !dbg !51
                                                                                                                                             %52 = getelementptr inbounds i32, ptr %c, i64 %10, !dbg !49
                                                   call void @llvm.aarch64.sve.st1.scatter.sxtw.nxv4i32(<vscale x 4 x i32> %62,
                                                                                                                                             store i32 %51, ptr %52, align 4, !dbg !50, !tbaa !43
                                                   ... <vscale x 4 x i1> %59, ptr %arrayidx5, <vscale x 4 x i32> %30), !dbg !51
                                                                                                                                             %53 = insertelement <vscale x 4 x i1> undef, i1 %cmp1.not, i1 false
                                                   \%63 = \text{zext} < \text{vscale x 4 x i32} > \%30 \text{ to } < \text{vscale x 4 x i64} > , !dbg !51
                                                                                                                                             %54 = insertelement <vscale x 4 x i1> %53, i1 %cmp1.not.headerCopy.1, i1 true
                                                   call void @llvm.aarch64.sve.st1.scatter.sxtw.nxv4i32(<vscale x 4 x i32> %30,
                                                                                                                                             %55 = insertelement <vscale x 4 x i1> %54, i1 %cmp1.not.headerCopy.1.2, i1
                                                   ... <vscale x 4 x i1> %59, ptr %arrayidx5, <vscale x 4 x i32> %30), !dbg !51
                                                                                                                                             ... false
                                                   %64 = insertelement <vscale x 4 x i1> undef, i1 %cmp1.not, i1 false, !dbg !51
                                                                                                                                             %56 = insertelement <vscale x 4 x i1> %55, i1 %cmp1.not.headerCopy.1.2.3, i1
                                                   %65 = insertelement <vscale x 4 x i1> %64, i1 %cmp1.not.headerCopy.1, i1
                                                                                                                                             ... true
                                                                                                                                             %57 = trunc i64 %10 to i32
                                                    ... true, !dbg !51
                                                                                                                                             %58 = call <vscale x 4 x i32> @llvm.aarch64.sve.index.nxv4i32(i32 %57, i32 1)
                                                    %66 = insertelement <vscale x 4 x i1> %65, i1 %cmp1.not.headerCopy.1.2, i1
                                                                                                                                              br label %new.latch
                                                    ... false, !dbg !51
                                                   %67 = insertelement <vscale x 4 x i1> %66, i1 %cmp1.not.headerCopy.1.2.3, i1
                                                    ... true, !dbg !51
                                                   %68 = trunc i64 %10 to i32, !dbg !51
                                                   %69 = call <vscale x 4 x i32> @llvm.aarch64.sve.index.nxv4i32(i32 %68, i32
                                                    .. 1), !dbg !51
                                                   br label %new.latch, !dbg !51
                                                         new.latch:
                                                          \%70 = \text{phi} < \text{vscale x 4 x i32} = [\%69, \%if.then], [\%11, \%linearized]
                                                          \%71 = \text{phi} < \text{vscale x 4 x i1} > [\%67, \%if.then], [\%12, \%linearized]
                                                          %72 = phi <vscale x 4 x i32> [ %58, %linearized ], [ %13, %if.then ]
```

CFG for 'foo' function

%73 = phi < vscale x 4 x i1 > [%56, %linearized], [%46, %if.then]

br label %for.body